ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Monthly Peer Reviewed Open Access International e-Journal

CloudDiag for Production Cloud Computing Environment

M. Chiranjeevi
M.Tech Student,
Department of Computer Science & Engineering,
Narayana Engineering College Gudur.

V. Ananth Krishna, M.Tech, Ph.D,
HOD,
Department of Computer Science & Engineering,
Narayana Engineering College Gudur.

Abstract:

Performance diagnosis is labor intensive in production cloud computing systems. Such systems typically face many realworldchallenges, which the existing diagnosis techniques for such distributed systems cannot effectively solve. An efficient,unsupervised diagnosis tool for locating fine-grained performance anomalies is still lacking in production cloud computing systems.This paper proposes CloudDiag to bridge this gap. Combining a statistical technique and a fast matrix recovery algorithm, CloudDiagcan efficiently pinpoint fine-grained causes of the performance problems, which does not require any domain-specific knowledge to thetarget system. CloudDiag has been applied in a practical production cloud computing systems to diagnose performance problems.

Index Terms: Cloud computing, performance diagnosis, request tracing

I.INTRODUCTION:

PERFORMANCE diagnosis is labor intensive, especially fortypical production cloud computing systems. In suchsystems, a lot of software components bear a large number ofreplicas (component instances) distributed in differentphysical nodes in the cloud. They can be assembled intomultiple types of services, serving large amounts of userrequests.

The services provisioned by the cloud are oftenprone to various performance anomalies (e.g., SLA violations[1]) caused by software faults, unexpected workload, orhardware failures. Such defects may, however, be manifestedonly in a small part of component replicas, hiding themselvesin a large number of normal component replicas.

Our experiences in performance diagnosis for Alibaba-Cloud Computing1 show that troubleshooting performanceanomalies in practical production cloud computing systems faces many real-world challenges.It is very difficult to applyexisting diagnosis techniques for such distributed systems.We summarize the new design challenges as follows:

1. Performance diagnosis in fine granularity. A componenttypically has alot of replicasin a production cloudsystem. Within one component, there are manyperformance-related private methods (i.e., those invokedinsidethe component) and public methods (i.e.,the interfaces invoked by other components). It is verychallenging to localize anomalous methods as well astheir corresponding physical replicas. Current approachesgenerally focus on locating anomalousphysical nodes (e.g., [2]) or logical components (e.g.,[3]).

2. Unsupervised performance diagnosis. Many existingperformance diagnosis techniques resort to systembehavior models in identifying anomalies [4], [5]. Unfortunately, it is hard to manually build suchmodels in production cloud systems, given theircomplexity in system scale. In addition, cloud servicesare generally composed of many components developedby different teams, which are independentlyupdated online.

3. Performance diagnosis with high efficiency. Copingwith large runtime data generated by a production cloud system efficiently is a challenging task inperformance diagnosis.This paper bridges this gap by proposing CloudDiag.CloudDiag periodically collects the end-to-end tracing data (In particular, execution time of method invocations) fromeach physical node in the cloud. It then employs acustomized Map-Reduce algorithm to proactively analyzethe tracing data.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Monthly Peer Reviewed Open Access International e-Journal

Specifically, it assembles the tracing data ofeach user request, and classifies the tracing data intodifferent categories according to call trees of the requests. When the cloud system is suffering performancedegradation (e.g., average response time of user requests is larger than a threshold), a cloud operator can accessCloudDiag with its web interfaces to conduct a performancediagnosis. With the request tracing data, CloudDiag willperform a fast customized matrix recovery algorithm toinstantly identify the method invocations (together with thereplicas they locate) which contribute the most to theperformance anomaly. The whole process requires no domain-specific knowledge to the target service.CloudDiag has been successfully launched in diagnosingperformance problems for the production cloud systems inAlibaba Cloud Computing. We report three case studies in our real-world performance diagnosis experiences to demonstratethe effectiveness of CloudDiag in helping the operatorslocalize the primary causes of performance problems.

II.RELATED WORK:

2.1 Preliminaries:

A typical production cloud (e.g.,, the data-centric cloud computing facilities offered by Alibaba Cloud Computing for Alibaba Inc.) generally offers a lot of concurrent services. Services work collaboratively to support a cloud application. For example, an e-mail application hosted in Ablibaba Cloud Computing is supported by many services that handle the e-mail-relevant operations such as sending an e-mail, loading an e-mail, and listing emails.

From a service-oriented perspective, a service in the cloud is for handling a certain type of user requests (e.g., reading an email). A service is typically composed of many components. Each component often contains a large number of replicas (component instances) distributed in different physical nodes in the cloud for fault-tolerance, load balancing, and elasticity considerations.

2.2 Framework of CloudDiag:

Performance anomalies in cloud systems will manifest themselves as anomalous response time of user requests.

Since a service is composed of a lot of components, a service with anomalous performance must have involved some components with performance anomalies. A component typically has a lot of replicas in a production cloud system; however, the performance anomaly of a component may be manifested only in a small part of its replicas. This will cause the performance degradation of the involving service, which is frequently observed in the cloud computing systems of Alibaba Inc. Fig. 1 shows the execution time of a component method in different replicas in a 100-node cloud system. We can instantly see that only a small part of replicas (e.g., nodes 8 and 12) are anomalous when executing the method.

CloudDiag is composed of three major parts, i.e.,

1) collecting the performance data; 2) assembling the performance data; and 3) identifying the primary causes of the anomalies. We briefly overview each part as follows:Collect performance data CloudDiag traces user requests at a given sampling rate to expose performance data. For the sampled requests, each component replica records the performance data and saves them in its local storage. An important consideration is what kind of performance data CloudDiag should collect and how. CloudDiag adopts an instrumentation-based approach that collects the execution time of each component method. Details are discussed in Section 3. Assemble performance data. CloudDiag should first assemble the performance data distributed in numerous component replicas in a request-oriented way.

In other words, the performance data belonging to the same requests are correlated together. CloudDiag will then analyze such request-oriented performance data and infer the call tree of each sampled request. A customized map-reduce process is utilized to group requests into different categories based on their call trees. Requests within one category share the same call tree.

Identify the primary causes of anomalies. CloudDiag then identifies the anomalous categories according to their latency distribution. Then, for each anomalous category, a fast customized matrix recovery algorithm (i.e., robust principal component analysis (RPCA) [11]) is employed to identify the anomalous method invocations together with the replicas they are located.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Monthly Peer Reviewed Open Access International e-Journal

Details are discussed in Section 4. Note that Steps 1 and 2 are relatively time-consuming tasks since they work on the massive tracing data generated by the entire production cloud system. Hence, for efficiency considerations, CloudDiag performs these two steps proactively. In other words, CloudDiag conducts the tracing data collection and assembly during the execution of the system. All categories are stored in a BigTable-like storage system [12] for further performance diagnosis when performance anomalies are detected.

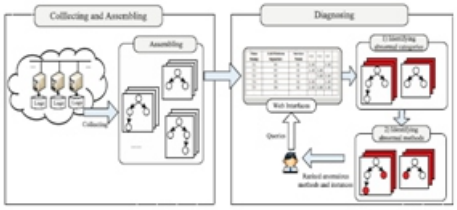

Fig. 1. A System overview of CloudDiag

III. PERFORMANCE DATA COLLECTION:

In this section, we introduce what kind of performance data that CloudDiag should collect and how to collect them. Our instrumentation-based tracing approach will produce performance data when a sampled request is being processed in each component replica. Specifically, each component method, when being invoked or returning, will generate a log entry.

The data structure of a tracing log entry is shown in Fig. 3a, which contains five items. Host indicates the machine where the component replica locates. Time stamp records the time of the event occurrence (i.e., a method invocation or a method return). RequestID is the global identifier of a request. MID is a unique identifier for request tracing purpose, which will be discussed later. The Method field saves the name of the method invoked. Lastly, Flag indicates whether this is a method invocation or a method return. Fig. 3a also shows two example log entries that record the invocation of the AliStorage.readFile method and its return.

Host	Timestamp	RequestID	MID	Method	Flag
Host1	2012-01-01 16:31:31.690272	169	739	AliStorge.ReadFile	Start
Host1	2012-01-01 16:31:32.991376	169	739	AliStorge.ReadFile	End

(a) Log entry for method execution time

Host	Timestamp	RequestID	Caller MID	Callee MID
Host1	2012-01-01 16:31:31.690724	169	739	991

Fig. 2. Tracing log formats and examples

RequestID should be unique for every request. It is assigned when a request arrives the system. Typically, a cloud service may have multiple entry nodes for the same type of requests. To guarantee the uniqueness of RequestID, an entry node will assign the RequestID (a unique 64-bit integer) as the concatenation of two integers: One is the unique number to identify the entry node p per se, and the other is incremental with each new request.

To guarantee the order of the invoked methods, we design hierarchical identifiers to trace a request. Specifically, before a node calls a method in another node, besides passing the RequestID to the callee, the caller also generates an MID, a unique integer, for the callee. When a request enters the first component method in the system (i.e., the request entry method), the MID of the method is initially set identical to the RequestID. CloudDiag then records the MIDs of the caller and the callee in the logs of the caller as well.Fig. 3b shows such a log format.Thus, CloudDiag can recover the caller-callee relationships according to the MIDs. The order of method invocations can then be correctly recovered and an entire performance trace of a request can, thus, be obtained. Fig. 5 shows an example of combining the tracing logs distributed over three hosts and retrieving the call tree.

IV. DIAGNOSING ANOMALIES WITHOUT DOMAIN KNOWLEDGE:

In this section, CloudDiag first employs a statistical technique to detect anomalous categories that contain latency-anomalous requests. Then, from anomalous categories, a fast matrix recovery algorithm, namely, RPCA, is adopted to identify the anomalous methods and instances. Details are as follows: 4.1 Identifying Anomalous Categories We cannot rely only on the response latency of a request to check whether a request is anomalous. Long response latency does not indicate a failure. For example, the response latency of one request reading a file from hard disk is several times longer than that of another reading a file from cache.

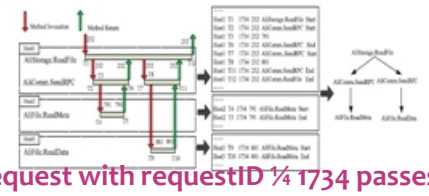

Fig. 3. A request with requestID ¼ 1734 passes through three hosts.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Monthly Peer Reviewed Open Access International e-Journal

When the SendRPC method at Host 1 invokes the ReadMeta method at Host 2 and the ReadMeta method at Host 3, it will generate the MIDs (791 and 801) for the two callees. CloudDiag can then recover the caller-callee relationships according to the third and sixth log entries of Host 1.

4.2 Identifying Anomalous Methods:

In an anomalous category of requests, our aim now is to pinpoint the anomalous method invocations that are responsive for the performance anomaly of the requests. For such a category of requests, we can create an m n matrix M, where n is the number of the invoked methods in the corresponding call tree and m is the number of the requests that bear the same call tree. Mij denotes the execution time of the jth method when depth-first traversing the call tree of the ith request. Column MðjÞ denotes the invocation time vector of the jth method. Intuitively, we can identify the anomalous method by measuring the execution time deviation of each method one by one. However, this cannot capture the correlations of the invoked methods, and will, hence, cause imprecise diagnosis results. Furthermore, such a statistical analysis can only identify anomalous methods, but cannot find out on which replicas the anomalous methods are executed.

Hence, we design an unsupervised machine learning algorithm to automatically learn the characteristics of the invoked methods and identify which methods are anomalous together with on which replicas they are executed. We discuss the details as follows: First of all, most requests with the same call tree bear similar performance data. In other words, most of the rows are correlated. As a result, the performance matrix M bears a low rank. Such a property is the basis of many existing approaches [16], [17], [18], [19] to widely employ principal component.

V. EVALUATION:

CloudDiag has been launched in Alibaba Cloud Computing Company to perform anomaly diagnosis in its production cloud computing systems. This section reports three case studies during our experiences in using CloudDiag in Alibaba. Our target cloud system is a cloud facility for Aliyun Mail, a production e-mail system that provides free e-mail service to the public.

2 ListMail, ReadMail, and SendMail are three services that are utilized to handle requests of listing mail titles, reading mail contents, and sending mails, respectively. They are the typical services of our target system, and are the focus of our experimental studies. Services are composed of a series of components (e.g., storage and communication). Each component has many homogeneous replicas that are deployed on different hosts. Currently, more than 20 million user requests are handled per day. On average, a request will typically go through over 10 hosts, invoking over 100 instrumented methods. By default, requests are sampled with the ratio of 1/200. Generally, the target cloud system would produce about 30-50 gigabytes (around 120-200 million lines) of tracing logs per hour.

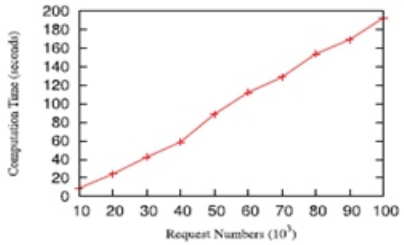

Fig. 4. Scalability of the RPCA based anomaly detectionCloudDiag is deployed in a small cluster with 10 nodes.

Each node is a typical low-end computer running Linux RHEL 5.4. CloudDiag proactively pulls the tracing data from the target cloud in a periodical manner (once every hour in our experiments). It then runs a customized mapreduce process to assemble and classify tracing data. First, map tasks assign correlated tracing logs that belong to the same requests to corresponding reduce tasks. Second, reduce tasks generate and classify requests into categories

5.1 Scalability Evaluation:

For efficiency consideration, CloudDiag is required to be scalable to the massive performance data. Since CloudDiag conducts the tracing data collection and assembly proactively, the anomaly diagnosing step is the only issue that will influence the scalability of CloudDiag. 5.2 Evaluation of Diagnosing Results In this section, we demonstrate how CloudDiag helps operators detect real-world performance anomalies that happened in Alibaba cloud computing platform. We adopt the following two measures to evaluate the effectiveness of CloudDiag. The first is precision ¼ T P TPþFP ,

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Monthly Peer Reviewed Open Access International e-Journal

which measures the exactness of our approach. The second is recall ¼ T P T PþFN , which measures the completeness. TP refers to the number of true positives (i.e., the number of anomalous methods); FP refers to the number of false positives (i.e., the number of normal methods that are mistaken for the anomalous); FN refers to the number of false negatives (i.e., the number of anomalous methods that are mistaken for the normal).

5.2 Evaluation of Diagnosing Results:

In this section, we demonstrate how CloudDiag helps operators detect real-world performance anomalies that happened in Alibaba cloud computing platform. We adopt the following two measures to evaluate the effectiveness of CloudDiag. The first is precision ¼ T P TPþFP , which measures the exactness of our approach. The second is recall ¼ T P T PþFN , which measures the completeness. TP refers to the number of true positives (i.e., the number of anomalous methods); FP refers to the number of false positives (i.e., the number of normal methods that are mistaken for the anomalous); FN refers to the number of false negatives (i.e., the number of anomalous methods that are mistaken for the normal). anomalies if a black-box tracing mechanism is applied. To incorporate black-box tracing mechanisms with CloudDiag, a future direction is to explore black-box tracing mechanisms so that a fine granularity (i.e., in method invocation level) can be achieved. To this end, the runtime instrumentation (e.g., [30]) can be a promising technique.

In conclusion, this paper proposes CloudDiag, an efficient, unsupervised diagnosis tool for locating fine-grained performance anomalies. The experimental results demonstrate that our approach scales well to massive tracing data. We also report our experiences that CloudDiag can effectively and conveniently help operators diagnose three real-world performance problems with high precision and recall. Extensive work has employed explicit annotation-based instrumentation to conduct performance monitoring, tuning, and debugging for distributed systems, which is surveyed as follows: Magpie [23] applies application-specific event schemas to correlate the resource consumption of individual requests with the goal of understanding performance. Pip [4] and Ironmodel [24] compare users' actual behavior with self-defined expectation to determine whether a request is anomalous or not.

It is very hard to construct these models because they require much specific domain knowledge. Compared to them, CloudDiag considers the intrinsic characteristics of request latencies to determine the anomalous-method invocations, which requires no specific domain knowledge.

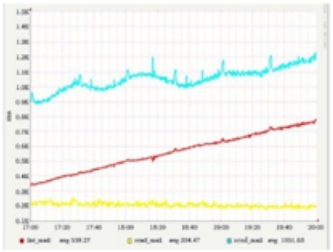

Fig. 5. The average latency of ListMail service increases nearly by 1/2 in about 3 hours.

CONCLUSION:

Pinpoint [13] traces request call relationship in multilayers of web service components and adopts a clustering algorithm to group failure and success logs. It can only find out the anomalous components. In comparison, CloudDiag can identify the latency-anomalous methods together with corresponding physical replicas. Dapper [25] introduces an infrastructure of performance monitoring. It keeps tracing logs into Bigtable [12]. Yet this approach does not describe how to use these logs to diagnose performance problems. Spectroscope [7] aims to find the primary cause of performance changes between two time periods, while our work does not assume the existence of a "correct" time period when the system performs normally. P-Tracer [26] can be utilized to identify the performance anomalies that manifest themselves as the change in the ratios of the chosen call trees, while CloudDiag can localize the latency-anomalous methods within call trees..

REFERENCES:

[1] V. Emeakaroha, M. Netto, R. Calheiros, I. Brandic, R. Buyya, and C. De Rose, "Towards Autonomic Detection of SLA Violations in Cloud Infrastructures," Future Generation Computer Systems, vol. 28, pp. 1017-1029, 2011.

[2] Z. Lan, Z. Zheng, and Y. Li, "Toward Automated Anomaly Identification in Large-Scale Systems," IEEE Trans. Parallel and Distributed Systems, vol. 21, no. 2, pp. 174-187, Feb. 2010.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Monthly Peer Reviewed Open Access International e-Journal

[3] H. Malik, B. Adams, and A. Hassan, "Pinpointing the Subsystems Responsible for the Performance Deviations in a Load Test," Proc. IEEE 21st Int'l Symp. Software Reliability Eng. (ISSRE), pp. 201-210, 2010.

[4] P. Reynolds, C. Killian, J. Wiener, J. Mogul, M. Shah, and A. Vahdat, "Pip: Detecting the Unexpected in Distributed Systems," Proc. USENIX Third Symp. Networked Systems Design and Implementation (NSDI), pp. 115-128, 2006.

[5] E. Thereska and G. Ganger, "Ironmodel: Robust Performance Models in the Wild," ACM SIGMETRICS Performance Evaluation Rev., vol. 36, no. 1, pp. 253-264, 2008.

[6] E. Thereska, B. Salmon, J. Strunk, M. Wachs, M. Abd-El-Malek, J. Lopez, and G. Ganger, "Stardust: Tracking Activity in a Distributed Storage System," ACM SIGMETRICS Performance Evaluation Rev., vol. 34, no. 1, pp. 3-14, 2006.

[7] R. Sambasivan, A. Zheng, M. De Rosa, E. Krevat, S. Whitman, M. Stroucken, W. Wang, L. Xu, and G. Ganger, "Diagnosing Performance Changes by Comparing Request Flows," Proc. USENIX Eighth Symp. Networked Systems Design and Implementation (NSDI), pp. 43-56, 2011.

[8] A. Chanda, A. Cox, and W. Zwaenepoel, "Whodunit: Transactional Profiling For Multi-Tier Applications," ACM SIGOPSOperating Systems Rev., vol. 41, no. 3, pp. 17-30, 2007.

[9] R. Fonseca, G. Porter, R. Katz, S. Shenker, and I. Stoica, "X-Trace: A Pervasive Network Tracing Framework," Proc. USENIX Third Symp. Networked Systems Design and Implementation (NSDI), pp. 271-284, 2007.

10] M. Chen, A. Accardi, E. Kiciman, J. Lloyd, D. Patterson, A. Fox, and E. Brewer, "Path-Based Faliure and Evolution Management," Proc. USENIX Symp. Networked Systems Design and Implementation (NSDI), pp. 23-36, 2004.

[11] E. Candes, X. Li, Y. Ma, and J. Wright, "Robust PrincipalComponent Analysis?" Arxiv Preprint arXiv:0912.3599, 2009.

[12] F. Chang, J. Dean, S. Ghemawat, W. Hsieh, D. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. Gruber, "Bigtable: A Distributed Storage System for Structured Data," ACM Trans. Computer Systems, vol. 26, no. 2, pp. 1-26, 2008.

[13] M. Chen, E. Kiciman, E. Fratkin, A. Fox, and E. Brewer, "Pinpoint: Problem Determination in Large, Dynamic Internet Services," Proc. IEEE Int'l Conf. Dependable Systems and Networks (DSN), pp. 595-604, 2002.

[14] D. Mills, "Network Time Protocol (Version 3) Specification, Implementation and Analysis," 1992.

[15] H. Abdi, "Coefficient of Variation," Encyclopedia of Research Design, N. Salkind, ed., pp. 1-5, SAGE, 2010.

[16] W. Xu, L. Huang, A. Fox, D. Patterson, and M. Jordan, "Online System Problem Detection by Mining Patterns of Console Logs," Proc. IEEE Int'l Conf. Data Mining (ICDM), pp. 588-597, 2009.

[17] A. Oliner and A. Aiken, "Online Detection of Multi-Component Interactions in Production Systems," Proc. IEEE/IFIP 41st Int'l Conf. Dependable Systems and Networks (DSN), pp. 49-60, 2011.

[18] H. Ringberg, A. Soule, J. Rexford, and C. Diot, "Sensitivity of PCA for Traffic Anomaly Detection," ACM SIGMETRICS Performance Evaluation Rev., vol. 35, no. 1, pp. 109-120, 2007.

[19] H. Mi, H. Wang, G. Yin, H. Cai, Q. Zhou, and T. Sun, "Performance Problems Diagnosis in Cloud Computing Systems by Mining Request Trace Logs," Proc. IEEE Network Operations and Management Symp. (NOMS), pp. 893-899, 2012.

[20] I. Jolliffe, Principal Component Analysis. Springer, 2002.

[21] Z. Lin, M. Chen, L. Wu, and Y. Ma, "The Augmented Lagrange Multiplier Method for Exact Recovery of Corrupted Low-Rank Matrices," Arxiv preprint arXiv:1009.5055, 2010.

Author Details:

Mr. M. Chiranjeevi
is a student of Narayana Engineering College,Gudur, presently he is pursuing M.Tech (C.S) and his project onCloudDiag for Production Cloud Computing Environment.

Mr. V. Ananth Krishna,
excellent teacher. Hereceived M.Tech, Ph.D. and working as H.O.D for the department of Computer Science & Engineering in Narayana Engineering College Gudur.